A COMPUTATIONAL MODEL FOR SOLVING RAVEN'S PROGRESSIVE MATRICES INTELLIGENCE TEST

A Dissertation
Presented to
The Academic Faculty

By

Snejana Shegheva

In Partial Fulfillment
of the Requirements for the Degree
of Master of Science in the
School of Computer Science

Georgia Institute of Technology

August 2018

A COMPUTATIONAL MODEL FOR SOLVING RAVEN'S PROGRESSIVE MATRICES INTELLIGENCE TEST

Approved by:

Dr. Ashok Goel, Advisor
College of Computing
Georgia Institute of Technology

Dr. Keith McGreggor
College of Computing
Georgia Institute of Technology

Dr. David Joyner
College of Computing
Georgia Institute of Technology

Date Approved: November 30, 2017

If understanding language and other phenomena through statistical analysis does not count as true understanding, then humans have no understanding either.

Ray Kurzweil

To Michael Maldonado who opened my eyes to the unseeable and opened my mind to the unbelievable.

ACKNOWLEDGEMENTS

The inspiration for building a computational model for solving Raven's Intelligence test started during the summer of 2015 when I took a Knowledge Based Artificial Intelligence (KBAI) class taught by Dr. David A. Joyner. After finishing the class I wished to continue deepening my knowledge in field of cognition and Dr. Joyner suggested further exploring computational modeling for the Raven's test. That proved to be a turning point in my research interests. A two years and many discarded hypotheses later, the Structural Affinity method was born. The orchestration of mathematics, data science and computer science produced a theory of intelligent problem solving that convinced me of the importance of statistical thinking in understanding human problem solving process. Many building blocks of the computational model were borrowed from the KBAI class. Each Raven's problem was already pre-processed - scanned and divided into separate images to feed into the visual algorithm that reduced the potential frustration with battling the infrastructure code.

My enormous gratitude goes to Dr. Ashok Goel who carefully guided the research in a direction that continued to raise very challenging questions about the knowledge representation, interpretation of the proposed hypotheses, and the claims about connections and similarities of the computational models with human problem solving. Discussions that took place during Design & Intelligence Lab Agency spawned multiple ideas, especially those related to the connection of Markov Graphical Models representation and the human cognitive processes.

I would like to additionally acknowledge all members of Design & Intelligence Lab (past and present) that helped me grow as a researcher, especially Tesca Fitzgerald who shared her experiences in paper writing and publication process. And, finally, I would like to express my gratitude to Dr. Keith McGreggor and Dr. Maithilee Kunda whose research in Raven's Progressive Matrices served as strongest foundation for my work, and a source of inspiration for trying new idea and challenging my imagination.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Graphical models offer techniques for capturing the structure of many problems in real-world domains and provide means for representation, interpretation, and inference. The modeling framework provides tools for discovering rules for solving problems by exploring structural relationships. We present the Structural Affinity method that uses graphical models for first learning and subsequently recognizing the pattern for solving problems on the Raven's Progressive Matrices Test of general human intelligence. Recently there has been considerable work on computational models of addressing the Raven's test using various representations ranging from fractals to symbolic structures. In contrast, our method uses Markov Random Fields parameterized by affinity factors to discover the structure in the geometric problems and induce the rules of Carpenter et al.'s cognitive model of problem-solving on the Raven's Progressive Matrices Test. We provide a computational account that first learns the structure of Raven's problem and then predicts the solution by computing the probability of the correct answer by recognizing patterns corresponding to Carpenter et al.'s rules. We demonstrate that the performance of our model on the Standard Raven Progressive Matrices is comparable with existing state of the art models.

In this report, we make a claim that visual intelligence tests such as Raven Progressive Matrices can be solved using minimal knowledge of the high-level concepts such as objects' classification and spatial relationship between objects. We raise and attempt to address research questions about the knowledge representation that provides a sufficient opportunity to capture a pattern in geometrical intelligence tests. Our main question about *designing an agent that can explain its reasoning while still providing accurate solutions* can be broken down into three phases:

- What type of knowledge representation offers an opportunity to capture a pattern?
- What organization of the knowledge representation units facilitates pattern extraction process?

- How can the pattern extraction facilitate the learning process and provide a solution that is explainable?

We show how a minimal representation facilitates pattern extraction process by proposing a method for organizing the representational units using the framework of probabilistic graphical models. By orchestrating techniques from mathematics, data science and computer science, we design an agent that can *explain* its responses by reducing the graphical models to minimal topologies that capture higher-level concepts such as *strategies* for solving given intelligence tests. And finally, we discuss the key takeaways about knowledge representation, structure discoveries, heuristic reasoning, and a possible connection to the cognitive thinking process.

CHAPTER 1
INTRODUCTION AND BACKGROUND

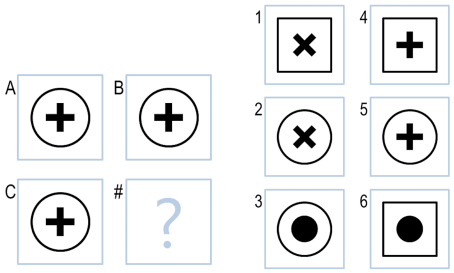
1.1 Introduction to Raven Progressive Matrices

Polya wrote that heuristic, or *ars inveniendi*, reasoning aims at discovering rules for solving problems for which an optimal solution may be impractical or requires a provisional plausible guess [1]. Newell has written that Polya's methods for problem-solving are directly relevant to mechanizing reasoning with computer programs [2]. However, heuristic reasoning traditionally has been used in conjunction with symbolic, propositional representations. With this work, we illustrate the use of heuristic reasoning with Markov Random Fields for addressing problems on tests of human intelligence. We aim at providing a complementary view on problem solving that exploits statistical reasoning over graphical representations of the problems on the Raven's test.

The need to be able to assess the degree of success for computational models of human cognitive processes has started an increasing trend of building computer systems capable of addressing tests of individual human intelligence [3]. Hernández-Orallo et al. present an extensive taxonomy of about thirty existing models varying in purpose, generalization and technology [4]. The diversity of of the problems ranges from geometric analogy [5] to odd-one-out [6, 7] and Raven's Progressive Matrices (RPM) [8, 9, 10, 11]. In this paper, we center our attention on RPM because its visual input, variety of problems, centrality in previous research, and a correlation with general human intelligence provide a suitable dataset for analyzing capabilities of a computational model that focuses on problem solving.

An RPM problem is a clever organization of geometric figures (see Figure 1.1 for an illustration). In part because of its simplicity and partly because of the high correlation with other measures of intellectual achievement [8], it is widely adopted in psychometrics,

the science of measuring intelligence and knowledge. The Standard Raven Progressive Matrices (SPM) test consists of five sets of twelve problems each, A through E, with the problems typically increasing in difficulty both within a set and across the sets. In this discussion, we will focus on SPM problems (both, 2x2 and 3x3) that are presented in black ink on white background. The left-hand side of the Figure 1.1 shows a 2x2 matrix similar to a problem from the SPM test, with the missing element in the lower right corner. The right-hand side shows a set of six possible answers for filling in the blank cell to complete the logical pattern in the matrix.



(a) RPM Problem Space (b) RPM Solution Space

Figure 1.1: Example 2x2 problem similar to one from the Standard Raven Progressive Matrices (SPM) test. (Due to copyright issues, all such figures in this paper illustrate problems *similar* to those on the SPM test.)

Early computational models of addressing RPM problems did not change the original problem representation, focusing instead on the rules needed to solve the problem. Carpenter et al, 1990, identified five distinct rules - ”constant in a row”, ”distribution of three values”, ”quantitative pairwise progression”, ”figure addition”, and ”distribution of two values”. Their method exemplified heuristic reasoning over propositional representations. The system proposed by Lovett et al. (2007) used prior knowledge of geometric elements and spatial relations to build qualitative spatial representations of the human-generated sketches of RPM problems, and then performing analogical reasoning on them. These models illustrate the *Analytic* strategy to solving Raven's problems [12].

More recently, two new approaches, *fractal* and *affine*, propose purely iconic visual rep-

resentations of the Raven's Progressive Matrices [13]. The *affine* method splits the matrix grid and the solutions grid into individual cells and performs affine transformations on the pixel representations. The *fractal* method takes this division further by partitioning the cells into fractal units and subsequently estimating similarity based on the features extracted from the fractal representations. Both approaches exemplify the *Gestalt* visual strategy, an alternative to the *Analytic* strategy [12]. An important aspect of these approaches is that they operate on a *transformed representation* of the original problem [4].

A more recent model in the *Analytic* approach analyzes RPM problems in terms of their practical difficulty as measured by the number of rules applied in Carpenter et al.'s model [14]. A Bayesian model in this tradition assigns prior probabilities to the rules in Carpenter et al.'s model to fit data on human performance on RPM problems [15]. Another anthropomorphic cognitive model [11] emphasizes problem-solving strategies evident among high-achieving human problem solvers.

We observe two core themes common to the above computational accounts: problem re-representation and exploitation of problem-specific heuristic strategies. We present an alternative computational method that combines the benefit of purely visual representation, problem re-representation, structural mapping, and heuristic reasoning nested in the problem-solving strategies. We use framework of Markov Random Fields (MRF) as the basis of the proposed mathematical representations because it enables us to express interactions between images in the RPM problems in a formal and very compact way. Markov Networks, which are a subclass of general graphical models, provide an advantageous mechanism for interpretation of the structure of the problem through assigning numerical values to interactions between its components.

1.2 Approach Motivation and Overview

The general process of our computational model is presented in Figure 1.2. The three main modules - *representation building*, *pattern learning* and *pattern recognizing* - constitute

the essence of the Structural Affinity computational model for solving Raven's Progressive Matrices. The low-level pixel information is extracted from images and represented as affinity factors which measure the compatibility between images. This information is used to create the next level of abstraction - a problem structure corresponding to a rule that most likely captures the logical sequence in the input image. The learned abstractions are stored in memory and later retrieved during the process of pattern recognition.

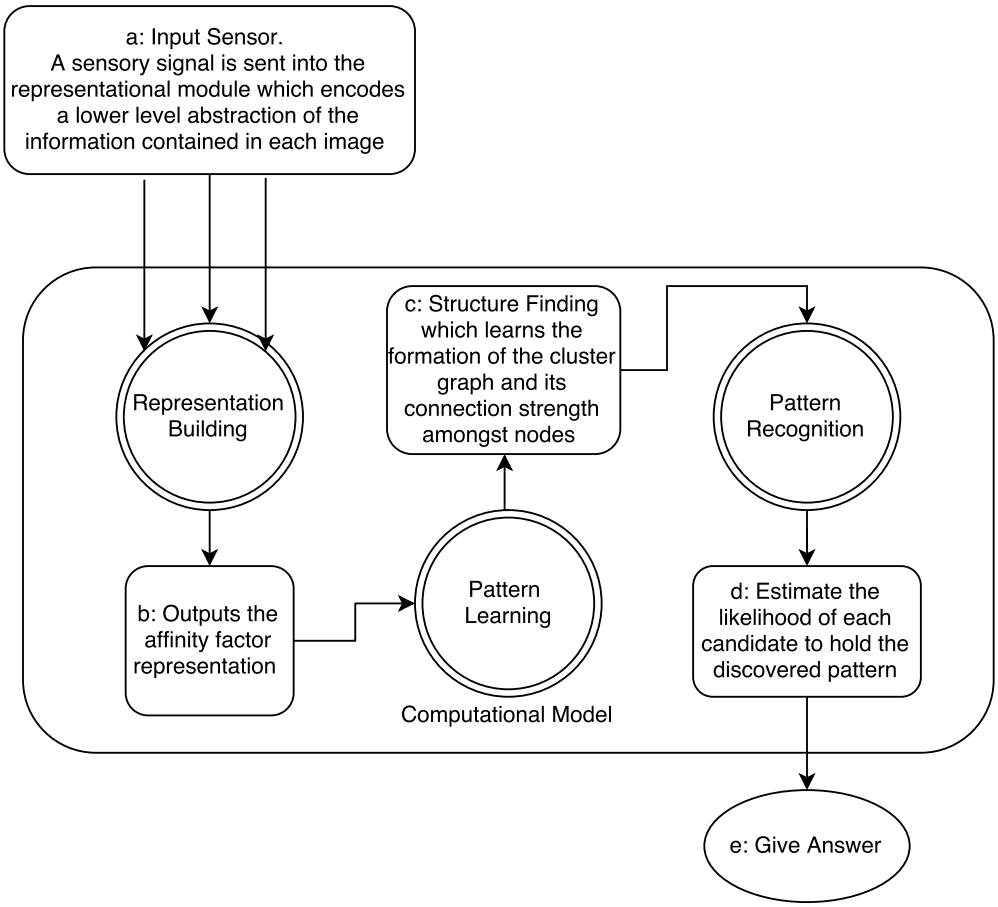


Figure 1.2: A diagram of the Structural Affinity computational model with three main modules - representation, pattern learning, and pattern recognizing.

We demonstrate the basic process on the example shown in Figure 1.3.

After receiving the image matrix (here, 3x3) and encoding the input with affinity factors, the pattern learning module starts the process of finding a structure which best encodes the spatial relationship between the images in the given problem. The search of a structure is initiated by analyzing the possible transformations when moving from left to right (row),

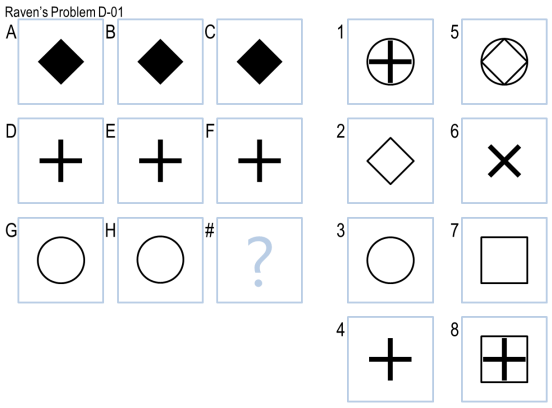


Figure 1.3: An example of the 3x3 Raven's problem

or from top down (column). Given that there may exist more than one structure that can encode the problem with sufficient soundness, our methods gives preference to the formations which satisfy the following two principles of parsimony:

- Minimal number of the nodes: our method seeks the minimal group of images which together represent a single unit of a pattern. For example, for the problem given in Figure 1.3, the smallest group contains three images, and we could have a few different ways of grouping - row-wise, column-wise, diagonal or triangular

- Feature invariance: of all candidate groups, our method gives preference to those clusters which undergo the minimal number of detected transformations. For example, as the row grouping (for Figure 1.3) preserves the identity property better than the column grouping, the row-structure provides a higher information gain on the pattern.

After the minimal structure is mapped out, the pattern recognition module collects the evidence of a pattern similarity by fitting each of the candidate solutions from the given list. The task of the pattern recognizer is to find the solution with highest likelihood. By substituting each candidate into the incomplete third row, the recognizer scores the resulting formation with the objective to maximize the identity function. This process arrives at the

correct solution by selecting the image that repeats the image observed in the third row - a circle. By being able to abstract the concept of a *rule* from the graphical representation of Raven's problem, the model learns to recognize a pattern. In the next section we show how to generalize this example by formalizing the method of exploiting the structure of the graph.

CHAPTER 2
TECHNICAL DETAILS OF THE WORK

2.1 The Structural Affinity Method

2.1.1 Knowledge Representation

The core challenge in solving a Raven's Intelligence test is identifying the logical pattern in a sequence of geometrical images, which when expanded optimally leads to a single correct answer. Thus, the issue here is to establish a representational structure parameterized in such a way that it directly correlates with the underlying pattern.

Our approach here is to model the interactions between images as undirected graph structure such as Markov Random Field. Each *cell* of the Raven's Matrix corresponds to a *node* variable in a network with edges capturing the interaction between variables. At the crux of the idea is the notion of *image affinity* which tracks an important measure for how compatible are the images within a group. The affinity between neighboring nodes is a *factor* function whose purpose is to parameterize the undirected graph without imposing a causal structure [16].

Let D be a set of N images. The affinity factor ϕ is then a function from $Val(D)$ to $\mathbb{R}$. As an illustration, let us consider the affinity factor for 2x2 Raven's Matrices. The smallest building block of the image is the pixel which we shall denote as x^1 or x^0 for white and black colors respectively. Table 2.1 shows one hypothetical factor function for a pair of images which requires four possible configurations of the pixels' color assignments.

For capturing level of agreement between two images, the factor function is takes a form of:

$$\phi(A,B) : Val(A,B) \mapsto \mathbb{R}^+ \tag{2.1}$$

Table 2.1: An example of the affinity factor ϕ for a pair of images A and B. It captures the interaction between variables by estimating the agreement between choice of pixel color. A high value for the assignments with matching pixel states, (a^0, b^0) and (a^1, b^1), correspond to a strong agreement, i.e. images A and B are very similar. On the other side, if the assignments with opposite pixel states, (a^0, b^1) and (a^1, b^0), are more likely indicated by a high ϕ value, then a significant transformation has been applied to image A to form an image B.

		$\phi(A,B)$
a^0	b^0	1000
a^0	b^1	200
a^1	b^0	500
a^1	b^1	2000

The affinity factor is calculated by counting the number of occurrences for each pixel configuration (a,b):

$$\phi_{(a,b)}(A,B) = \sum_{i=1}^{N} [C(A_i) = a] \wedge [C(B_i) = b] \tag{2.2}$$

where $C(X_i)$ is an operator function for reading a color bit information for pixel i of the image X represented by an array of N pixels.

A full Markov Network for the 2x2 Raven's Matrix is graphically visualized in Figure 2.1 with nodes encoding image labels and edges parameterized by affinity factor functions. For a 3x3 Raven's Matrix, the full graphical structure is more complex. However, we are mainly concerned with the factor function which takes a similar form with 2^3 possible assignments as indicated in Table 2.2.

Table 2.2: Affinity factor ϕ for a triple of images A, B and C.

			$\phi(A,B,C)$
a^0	b^0	c^0	ϕ_1
a^0	b^0	c^1	ϕ_2
a^0	b^1	c^0	ϕ_3
a^0	b^1	c^1	ϕ_4
a^1	b^0	c^0	ϕ_5
a^1	b^0	c^1	ϕ_6
a^1	b^1	c^0	ϕ_7
a^1	b^1	c^1	ϕ_8

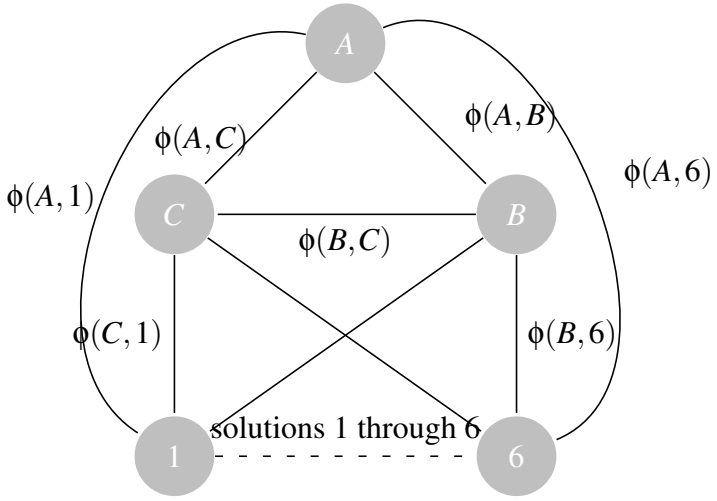


Figure 2.1: Markov Network for the 2x2 Raven's Matrix

2.1.2 Justification

Markov Networks are commonly used in research on computer vision for a variety of tasks such as image segmentation and object recognition [16]. By formulating a model with the ability to capture the interactions between neighboring images in an RPM problem, we can infer the logical pattern in a sequence of images. This method of representing an RPM problem as a Markov Network not only does not involve any propositional representations (such as shapes, objects, spatial relations), it does not even engage any image transformations (such as reflections, rotations, translations). Instead, the reasoning is based solely on the statistical interaction between pixels in the images.

2.1.3 Assumptions and Biases

An affinity function is biased towards preserving the geometric objects' shapes and color densities. Thus, the proposed computational model will suffer from performance loss on RPM problems where objects pass through a large number of transformations.

2.1.4 Structural Affinity Hypothesis

Solving Raven's problem using structural affinity method can be conceptualized as a process of going from a general model of the problem to a restricted Markov Network which captures the minimal amount of knowledge required for solving the problem. For the 3x3 RPM problem, the general model assumes inter-dependence of all components of the matrix space illustrated in Figure 2.2. For a graph representation, this condition requires $\frac{9(9-1)}{2} = 36$ edges which convolute the structure of the Raven's problem.

$$\begin{bmatrix} A & B & C \\ D & E & F \\ G & H & ? \end{bmatrix}$$

Figure 2.2: Raven Problem Matrix Space

For example, the objects found in the top left corner (denoted as A) control the objects in the remaining cells. Similarly, the objects in the cells B and D may directly influence the features of the objects in the cells C and G respectively. Following the pattern of this logic, we may conclude that the general model of the RPM is represented by a fully connected graph. The edges include influences of various strength between the objects leading to a complex network. This is problematic from a representational point of view as it hinders discovering the strategy for solving given Raven's problem. To address this issue, we restrict the number of the edges to the minimal set of the most influential connections that reduce the problem complexity and aid in discovering a structure. This constitutes the essence of the structural affinity method which we shall define more formally below.

Definition 2.1.1. Let $\mathbf{X}$ be a finite set of variables representing components in the visual problem (here, images of the Raven's test), and let $\phi_k(X_i, X_j)$ be a factor function that denotes the affinity between two variables. We define the *Structural Affinity* $\Psi(\mathbf{X})$ to be set

of factor functions that reveal the dependence structure in the full graph G:

$$\Psi(\mathbf{X}) : \{\phi_k(X_i, X_j) \,|\, X_i \not\perp\!\!\!\perp X_j \, and \, k \leq ||G||\} \tag{2.3}$$

where $||G||$ is the cardinality of the graph G given by the total number of edges.

With a reduced set of affinity factors we represent a relationship between the components of the Raven's problem which highlights the strongest interactions in the network (i.e., images X_i and X_j are not independent). The level of dependence is captured through the affinity factor which measures the degree to which the images are compatible with each other.

For example, the Raven's problem where the objects in a row simply repeat each other without any additional transformation, are said to be highly dependent resulting in high values of the affinity factor functions.

Backward Construction Process

One approach of picking an informative network structure is applying a *backward construction process*. Here, we start with a variable of interest, a candidate solution (denoted as ? in Figure 2.2), and iteratively estimate its dependence on the neighboring variables. Initially, all variables in the matrix spaces are considered to be neighbors with the solution item. We then prune the weak dependencies leaving only the minimal map of the network structure which may serve as a foundation for identifying a possible strategy for solving Raven's problem.

Below is the typical pseudo-algorithm for producing a minimal map based on independence test between variables in the graph. We augment the algorithm to reflect our domain - visual Raven's problem. The algorithm aims at discovering the set of neighboring nodes, also known as *Markov blanket*, which satisfy the requirement of a minimal map. The construction process starts with an arbitrary potential solution X to the Raven's problem. Throughout the process, variable X is subjected to independence tests with all cells in the

Algorithm 1: Pseudo-algorithm for building a minimal map

```
Data: Raven's problem image pixel data
Result: Markov Network structure
- Set initial network structure to an empty graph G;
- Select a candidate solution to the Raven's problem from the given list X;
for object U in the RPM matrix space do
    ϕ - compute affinity factor between variables X and U;
    t - independence test between variables X and U;
    if not independent then
        Add edge U - X to G;
    else
        // skip variable U
    end
end
for all parents of X do
    if parents are not independent then
        Add edge between parents to G;
    else
        // skip adding edge
    end
end
```

Raven matrix solutions. An edge is added if independence assumption is either not satisfied or significantly weaker then some experimentally defined threshold. The second order independence test is then applied to the neighboring variables which add edges between parents if there is strong evidence they influence each other.

Structural Affinity Hypothesis

The concept of the minimal map allows us to explore a possibility of identifying a correct solution of the Raven's problem by merely analyzing its structure as a mapping of strength between its components. Let's state the hypothesis of the Structural Affinity method, based on which we design an intelligent agent that predicts a solution to the Raven's problem:

> Given a Raven problem and its expected solution, the Markov network structure with the minimal map should have the highest structure score as compared to the networks with the sub-optimal solutions.

Under this hypothesis, solving Raven's Progressive Matrices intelligence test can be summarized with the following four steps of generating a set of structures and selecting the most likely one:

1. Create a hypothesis space Ω_m of all possible factor graphs, where m is number of possible solutions.
2. Define objective functions which optimize towards the desired properties of the graph.
3. Compute the scores for all resulting factor graphs.
4. Select the structure with the highest score as a problem solution.

The hypothesis Ω space is defined by creating a set of minimal structure maps for the Raven's problem with each candidate solution. The goal of the objective function is to quantify a particular property of a graph. For example, *maximum likelihood* objective function measures the fitness of the model to the data. Using the objective function, we then construct a search algorithm that attempts to find the network structure with the highest ranking score. We are not limited to a single objective function as various properties of the network can be best captured with a variety of scoring functions. Similarly to ensemble methods, the effect is either aggregated or filtered through a voting mechanism.

2.2 Pattern Learning with Rule Mapping

A presence of analytic strategy for solving Raven's problem is linked to the advanced reasoning skills in individuals taking the test [17]. Therefore, a computational model which goes beyond merely producing a solution is a good candidate for a cognitive account with capabilities to explain the strategy undertaken for solving the problem.

Various networks structures found during the minimal map searching lead to a natural interpretation of the outcome of the computational model reasoning. Both, the *presence* and the *strength* of the edges in the created network, provide insight on the nature of the masked dependencies. The resulting network topologies highlight properties of Raven's problem which may be connected to the problem structure through the set of rules described in Carpenter et al.'s seminal cognitive model of problem solving on SPM [8].

Constant Rule

The constant rule means that objects are repeated in one direction, but change throughout a different direction. For example, the test taker might notice that same objects appear in the row direction, therefore, substituting the missing element in the third row with an identical element. The test taker can apply the same reasoning if the constancy is observed in column direction instead.

What we have noticed is that network topologies, as shown in Figure 2.3 and Figure 2.4 provides a sufficient evidence of the constancy rule in the given Raven's problem. The minimal structure in the Figure 2.3, indicates that, if all three images in the third row are equally dependent on each other (weights of the edges are similar within a threshold), then it is a strategy equivalent to a *Constant in a Row* rule. Likewise, the structure shown in Figure 2.4 with the same topology, but with different nodes, recommends applying the *Constant in a Column* rule for solving the Raven's problem.

Distribution of Three Values Rule

Another network structure in the same family of topologies as those identified in constancy rules is a structure shown in Figure 2.5. By changing only the influential nodes, B and D, and leaving the rest of the structure intact, we couple the given topology with the *Distribution of Three Values* rule. The fact that this network belongs to the same family as constancy rules demonstrates the isomorphic property between these two rules. Distribu-

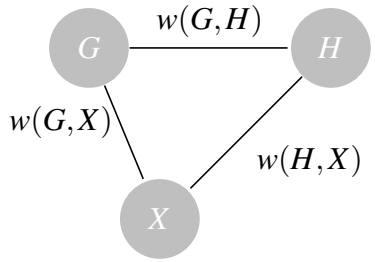


Figure 2.3: Network topology for constant in a row rule. The elements of the networks are images from the third row, G and H, and the missing element X from the solution space. All three nodes are connected with similar weight values indicating a repetition of objects in the row.

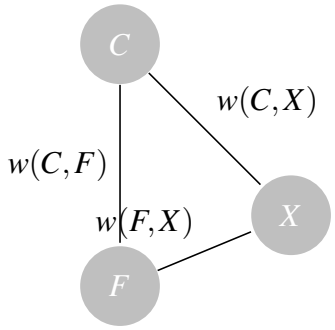


Figure 2.4: Network topology for constant in a column rule. The elements of the networks are images from the third column, C and F, and the missing element X from the solution space. Likewise as in the constant in a row rule, all three nodes are connected with similar weight values indicating a repetition of objects in the column

tion of three values is analogous to the constant rule where the direction of the dependency is not on a straight line, but, instead, it follows a *triangular* path as shown in Figure 2.6. This rule is also known as a *Permutation* rule since there should be one figure of each type in each column or row [15].

Figure Addition or Figure Subtraction Rule

A rule where an object in a third column is formed from either juxtaposition of the objects in the first two columns, or a subtraction of the objects of the second column from the first, is demonstrated in the example of Raven's problem in Figures 2.7 and 2.8, respectively. Griffiths et al. refer to these rules as *Logical OR* and *Logical AND* to classify the

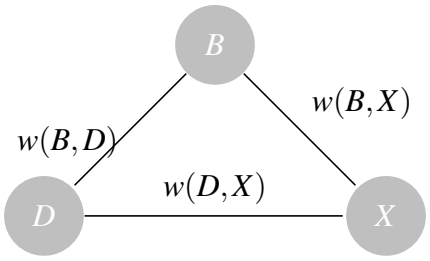


Figure 2.5: Network topology for distribution of three rule. The elements of the networks are images which follow a triangular path. This example uses images B and D, however, in a more general case, a triangle can be captured in few other combinations

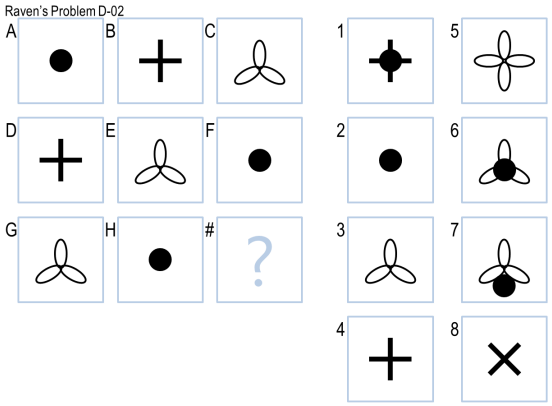


Figure 2.6: A RPM problem to illustrate the distribution of three values rule

transformation between figures as logical operations of disjunctions and conjunctions [15].

An *Addition in a Row* or *Addition in a Column* strategy are represented by topologies as shown in Figure 2.9 and Figure 2.10. The difference between the family of constancy and addition rules is the missing or much weaker link between the images on the third element (row or column) in the matrix space. Intuitively, this says that the objects from the two images in a row or column are independent from each other (or the dependence is weaker when compared to the other two edges), and are strongly linked through the third element which contains the features of both. i.e. juxtaposition.

The topologies for subtraction rules (see Figure 2.11 and Figure 2.12) differ from those of the addition rules. It includes the same exact nodes with swapping of the connections, exemplifying simple chain structures.

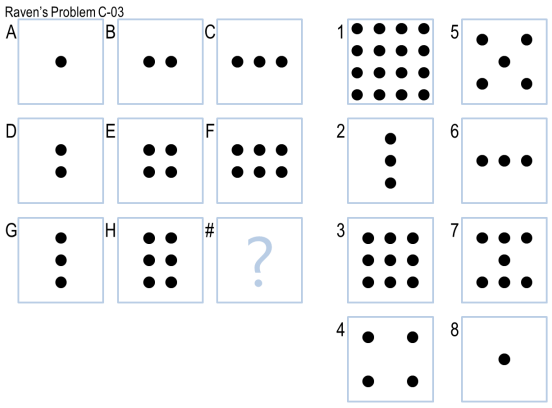


Figure 2.7: An RPM problem to illustrate the addition rule

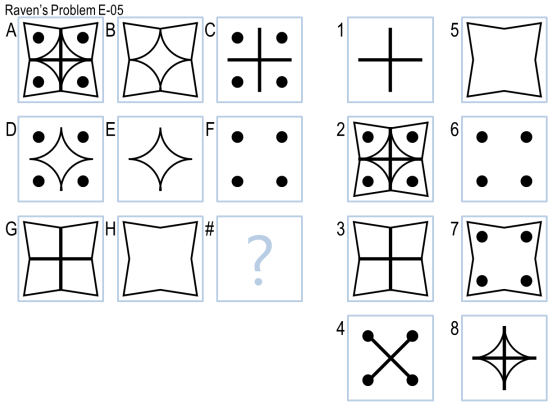


Figure 2.8: An RPM problem to illustrate the subtraction rule

Distribution of Two Values Rule

By continuing the pattern of permuting the edge which is dropped we arrive at the topologies shown in Figures 2.13 and 2.14. The rule of distribution of two, also known as *exclusive-or* reflects the logical XOR transformation between features of the images in RPM. An example of the RPM with this rule is shown in Figure 2.15 where the features of the third element in the row is constructed by removing common features between first two images.

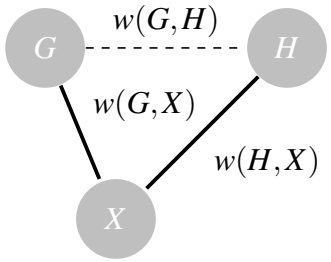


Figure 2.9: Network topology for the addition in a row rule. The elements of the networks are images from the third column, G and H, and the missing element X from the solution space. The nuance of this topology is that dependence between nodes G and H is not required.

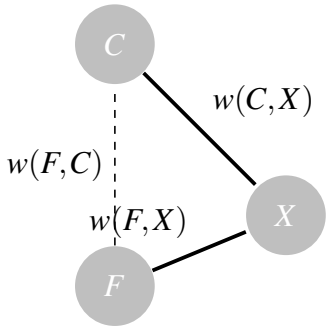


Figure 2.10: Network topology for the addition in a column rule. The elements of the networks are images from the third column, C and F, and the missing element X from the solution space. The dependency of the requirement between nodes C and F is relaxed similar to the addition in a row

Pairwise Progression Rule

Besides juxtaposing or subtracting objects, some Raven problems demonstrate a rule which requires reasoning about quantitative increments/decrements occurring through a change of size or quantity of the objects. Figure 2.16 illustrates a Raven problem with object resizing feature along the row and column directions. The graphical model which captures the minimal map of the progression rule is demonstrated in Figure 2.17.

2.2.1 Rule Inference

As the possible structure of the Raven's problem is mapped according to the topologies described above (a single problem can map to more than topology similarly to how it can

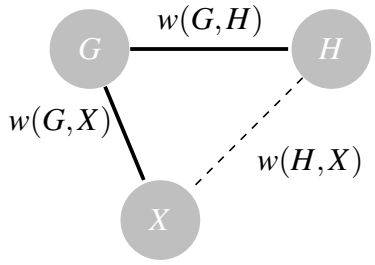


Figure 2.11: Network topology for the subtraction in a row rule. The elements of the networks are the same as in addition in a row, except that different edge is dropped from the network.

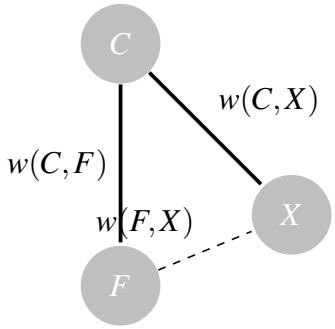


Figure 2.12: Network topology for the subtraction in a column rule. The elements of the networks are the same as in addition in a columns with a different edge being dropped

described by more than one rule), the Structural Affinity method enters the next phase by assigning a rule or a set of rules to the given problem. The assignment process here is akin to labeling the instances of observed topologies. Our objective here is to infer the likely rules given the observed structure represented by affinity factors over corresponding variables. D. Little et. al. has shown that a Bayesian model can successfully predict the

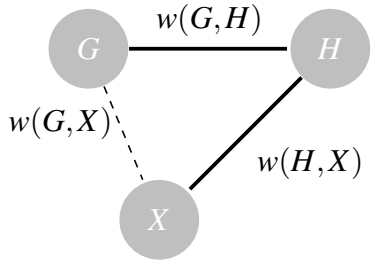


Figure 2.13: Network topology for the exclusive-OR in a row rule. The edge between outer nodes G and X is either dropped or of a weak strength

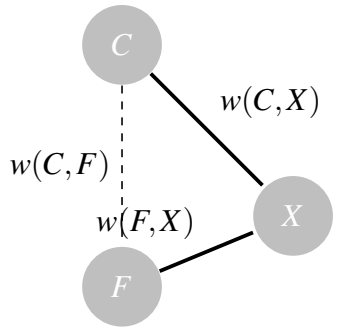


Figure 2.14: Network topology for the exclusive-OR in a column rule. The edge between outer nodes C and X is either dropped or of a weak strength

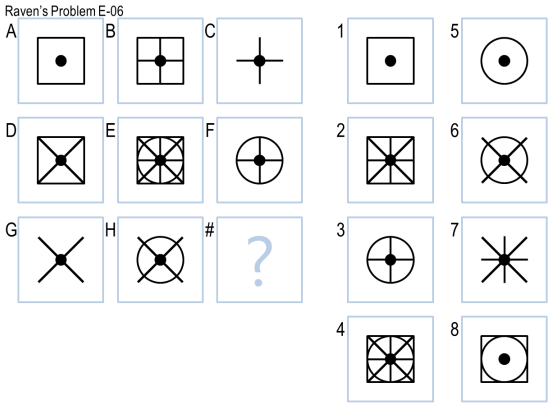


Figure 2.15: A RPM problem to illustrate the distribution of two values rule

rule by observing a collection of hand-coded features, those similar to Carpenter [15]. In our model, the handcrafting of features is not necessary as the rule is learned directly from the given problem. By following Little's et. al. Bayesian formulation, we compute the posterior probability of each rule using following formula:

$$P(g|s) \sim P(s|g) * P(g) \tag{2.4}$$

where $P(s|g)$ is the evidence for observing a structure s given rule g, and $P(g)$ is the prior for rule g. Here, $P(g|s)$ is an unnormalized measure since we use it for ranking purposes, and therefore do not need to convert it to actual probability in the range [0..1].

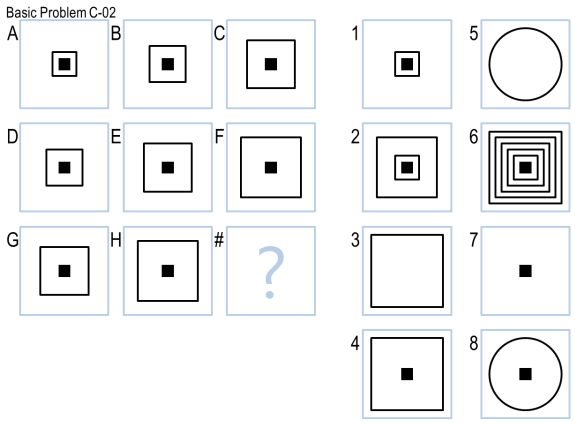


Figure 2.16: An RPM problem to illustrate the progression rule

Selecting the rule prior

When no additional knowledge is provided, it is typical to assume a **uniform** prior probability over the rule, i.e., each rule is equally likely to be seen *before* observing the structure. This assumption, although valid, especially in the context of no prior knowledge, has a clear limitation in expressing the varied frequency of each rule depending on the complexity of the problem. The easier rules such as *constancy* and *pairwise progression* are preferred to the *logical transformations* rules which appear on the more difficult sets [18]. To express this tendency, D. Little et. al. computed the frequency with which the rule appears in the study conducted by Carpenter and utilized that as more informative prior[15].

In our model, we use N. Georgiev's study results as a proxy to create a *Carpenter*-like prior which we manually classified as one of the instances of Carpenter rule given the unstructured description of the strategy for each Raven's problem in SPM set [19]. To provide more context to the pattern learning mechanism, we expanded the rule with the notion of directionality - row, column, diagonal, triangular, etc.

The table 2.3 shows both, the Uniform and Carpenter's prior for the rule with the context. In agreement with the Matzen et. al.s study, the most common rules are *pair progression* and *constancy* while logical AND and OR rules referred by Carpenter as subtraction

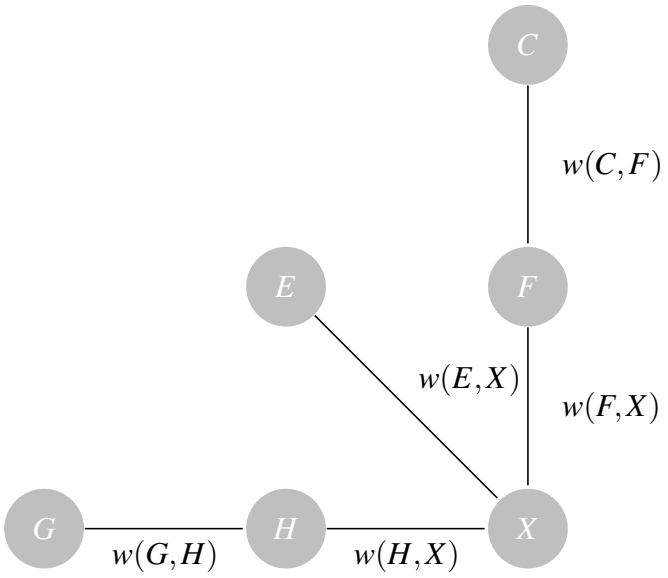


Figure 2.17: Network topology for the progression rule. The elements of the networks are third row, third column, and the diagonal element E. The dependency weakens as we travel from the bottom right to the top left corner of the problem.

and addition are among the least invoked strategies.

Computing the rule likelihood

We compute a rule likelihood by collecting evidence of the rule g from the observed structure s by performing a series of independence tests, such as χ^2, between affinity factors parameterizing a Markov network. The distribution of the χ^2 values within the network provides a basis for constraining functions $\mathcal{H}_i$ which after aggregation and normalization produce a score for the given combination of the structure s and rule g. The computed score measures the compatibility of the derived structure topology and the suggested rule:

$$P(s|g) = \frac{1}{Z}\sum_{k=1}^{n}\mathcal{H}_i(g) \tag{2.5}$$

where $P(s|g)$ is the likelihood of the structure s given rule g, Z is a normalization

Table 2.3: Uniform and Carpenter priors for each context-augmented rule

rule	uniform prior	carpenter prior
pairwise progression row	0.0667	0.1463
constant in a column	0.0667	0.122
constant in a row	0.0667	0.1098
distribution of three right	0.0667	0.0976
pairwise progression column	0.0667	0.0854
addition in column	0.0667	0.0732
addition in row	0.0667	0.0732
distribution of two	0.0667	0.0732
subtraction in column	0.0667	0.061
subtraction in row	0.0667	0.061
reflective symmetry column	0.0667	0.0366
reflective symmetry row	0.0667	0.0244
figure addition column	0.0667	0.0122
figure subtraction column	0.0667	0.0122
figure subtraction row	0.0667	0.0122

function, and $\mathcal{H}_i(g)$ is a constraining function of the form:

$$\mathcal{H}(g) = t(\bar{\chi^2}) \cdot w_t(g) \tag{2.6}$$

where t is statistics measure for the obtained χ^2 values of the independence tests and the $w_t(g)$ is the weight of the given statistics measure for the rule g.

$$w_t(g) = \begin{cases} > 0, & \text{for boosting the statistics} \\ 0, & \text{if statistics is not relevant for the rule g} \\ < 0 & \text{for penalizing the statistics} \end{cases} \tag{2.7}$$

Various weights and statistics are derived experimentally, based on the heuristics of the rule. For example, *constancy* rules give preference to the uniform distribution of the χ^2 values, i.e. all edges are connected with similar strength values, while *quantitative* rules penalize *evenness* and give preference to the configurations with a gradual decrease of strength.

2.3 Predicting the Response - Pattern Recognition

Once the rule tokens have been induced from Raven's matrix space for each problem, our method predicts the solution to the missing object as follows: For every inferred rule R_i for the problem at hand, rank each candidate solution X_j by computing the average value across a set of corresponding objective functions f_n imposed by the rule R_i.

$$E(X_j) = \frac{1}{||R||} \sum_{i=1}^{M} \sum_{n=1}^{N} f_n(R_i) \tag{2.8}$$

where $E(X_j)$ is the estimated likelihood of the solution X_j given set of objective functions f_n.

An example of an objective function for the *constancy* rule is:

$$maximize \quad F = \phi_{0,0}(I_1, I_2) \tag{2.9}$$

where I_1 and I_2 are variables in the Markov Network representing two images in the Raven's problem affinity for which is being estimated with the factor ϕ. Here, the factor function is computed over states $(0,0)$ (black pixels) to maximize the potential of identity property between images I_1 and I_2. A full list of objective functions is presented in the Appendix B.

Finally, the solution is selected by simply finding the candidate with the maximal value of the likelihood:

$$X = \arg\max j \in [1..J] E(X_j) \tag{2.10}$$

The objective functions exemplify the heuristic reasoning of our approach by introducing conditions under which the most likely solution should be found. Given the statistical nature of the algorithm, the model combines the signals from an ensemble of heuristic functions to predict the solution that fits the observations represented in the form of affinity

factors. The heuristics attempt to capture our expectations of the behavior between images in the Raven's test under the assumptions of the inferred rules. For example, by assuming that the best strategy for solving a specific Raven's problem is applying a conjunction rule in the row, we impose a kind of heuristics that simultaneously maximize the pairwise relationships between the candidate solution and the images from the third row. We achieve such heuristic constraint by applying the following objective functions:

$$max \quad F = \phi_{0,0,0}(I_1, I_2, I_3) \tag{2.11}$$

$$min \quad F = \phi_{0,1,0}(I_1, I_2, I_3) \tag{2.12}$$

$$min \quad F = \phi_{1,0,0}(I_1, I_2, I_3) \tag{2.13}$$

Initial inspection of the ranked candidate solutions indicated a strong propensity to split the predictions into two groups separated by a large margin - highly unlikely candidates which when selected would probably be attributed to a random choice, and plausible candidates, the incorrect answers amongst which are typically due to *Incomplete Correlate* errors in human performance on the Raven's test[20]. Our technique currently does not directly measure the confidence, however, given the probabilistic nature of the approach, the confidence is inferable via the computed score for each candidate.

CHAPTER 3
ANALYSIS OF RESULTS

3.1 Setup Description

We evaluate our computational model for sets B through E of the Standard Raven's Progressive Matrices (SPM) test. This is because the five types of rules described by Carpenter et al. do not apply to set A which relies heavily on textures and not geometric pattern learning. Images were digitized for consumption by our computational model by re-creating the original Raven's problem to reduce the artifact of scanning. The resulting collection does not remove the noise related to image alignment; however, the statistical nature of the algorithm *smoothed out* the impact on the accuracy. Please refer to Appendix *A: Data Processing* for technical details on image transformation.

3.1.1 Rule Inference

To evaluate the rule inference algorithm, we use the results of the study performed by Georgiev where he presented an analysis of the logical relations in Standard Progressive Matrices [19]. He demonstrated that different items have various strategies by examining how 506 Bulgarian high school students work out the relations in the administered Raven's test. For each item for sets C through E, Georgiev described the underlying strategy in the textual form. For example, the first problem in the set C is described as *Horizontally the figure remains the same, vertically a circle is added*. In our evaluation, this translates to two Carpenter rule tokens - a *constant* in a row, and a *quantitative progression* in a column. As the set B was not available in this format, we manually labeled the expected token applicable to each item in the set. This step introduced additional rule not available in the Carpenter's taxonomy - *symmetry* vertical and horizontal, or a *reflection* over x-axis

and y-axis, respectively. The gathered and transformed data provided a sufficient baseline to estimate the performance of our model which we describe in the next sections.

3.2 Rule Inference Results

We have defined a Structural Affinity method which infers the most likely minimal map of a graphical model, fit to individual Standard Raven's Progressive Matrices problem. In the discussion on the inference results we consider three aspects - the overall frequency of each rule; the rule frequency inside each Raven's Set; and comparison of the inferred results to a ground truth.

3.2.1 Summary of the rules

The overall frequency of each induced rule, according to the Carpenter's naming scheme is reported in Figure 3.1. The most common rule is *pairwise progression* which makes an intuitive sense given that the entire premise of the Raven's Matrices is identifying the most logical progression, and most problems involve either increment and decrement of certain attributes in the problem. We report the results with directional differentiation - row or column - to simplify the response prediction. When combined, the *addition* rule is second by popularity metric followed by the *distribution of three values* rule. Frequency of *rules* is comparable to that of the *addition* and *distribution of three*. By analyzing the relative frequency of each rule, we can infer difficulty of each rule as inversely proportional to its occurrence. *Subtraction* (conjunction) *distribution of two values* (exclusive-or) are considered more difficult and they occur less frequently.

Table 3.1 reports the aggregated frequency of each inferred rule by combining the occurrences of the rule type in either direction. The number of rule tokens applied to a problem varies from 1 to 4, with a most frequent case of 2 rules per problem. In addition to inducing the rule, the algorithm augments the results with the directional variable (row, column, diagonal, or triangular), however, for the analysis and comparison we only kept

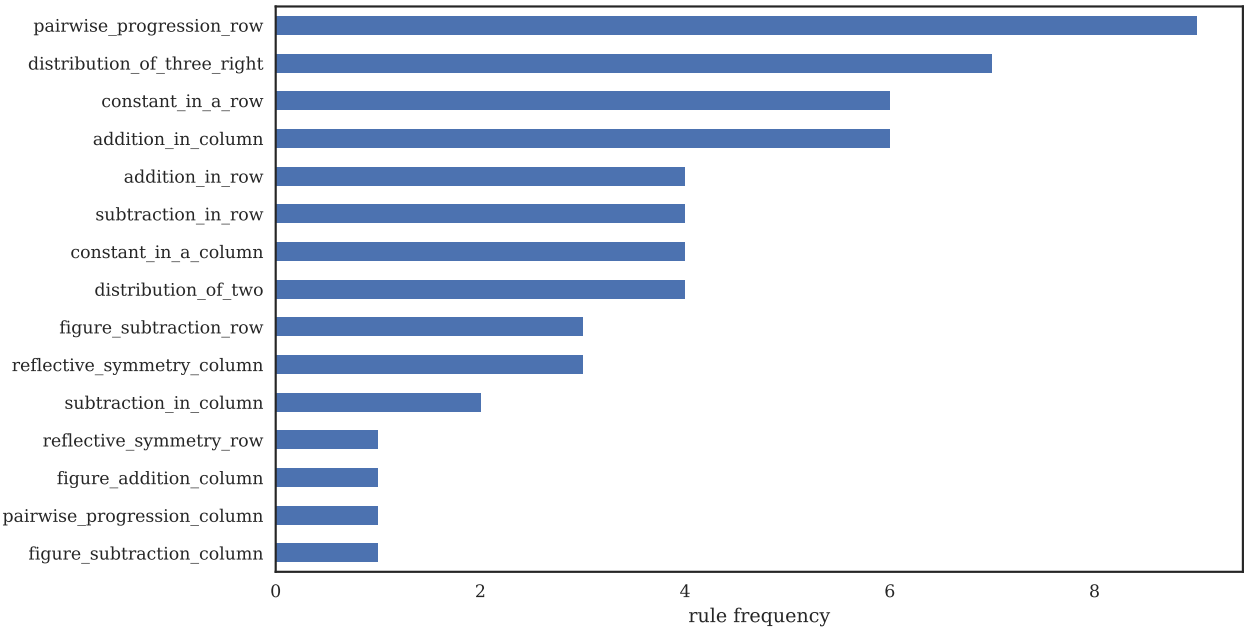


Figure 3.1: A distribution of rules induced from a Standard Raven's Progressive Matrices sets B through E*. Our evaluation differentiates between row and column direction to have a more accurate rule inference

* The rule induction algorithm for SPM is skipped for the set A which as it does not represent items requiring various rules.

the name of the rule.

It is interesting to note that each set is characterized by a different rule which occurs more frequently than others. The set C, which is also the easiest of the three, is primarily described by the *pairwise* rules. The set D involves a strategy of permuting the features ((distribution of three values)) which is slightly more difficult than extending quantitative increment/decrement. The set E demands to apply various logical transformations, such as XOR, AND, and OR which are amongst the most challenging rules [18].

3.2.2 Accuracy of the rules

To estimate the correctness of the rule induction outcome, we compared the algorithm response to a suggested logical relation in the items for each problem in sets C, D, and E [19]. For set B (which was not included in the Georgiev's analysis), we manually created the ground truth for each problem beforehand. Our performance measurements resulted in

Table 3.1: Induced Carpenter's rule for set B, C, D and E. We additionally included rule *Symmetry* for set B which is not classified as one the five Carpenter's rules. This analysis included cases where multiple tokens is required for solving a problem.

Rule				
	B	C	D	E
Constant in a Row or Column	6.0	6.0	7.0	0.0
Distribution of three values	0.0	0.0	8.0	0.0
Distribution of two values	0.0	0.0	0.0	1.0
Figure Addition or Subtraction	3.0	5.0	1.0	21.0
Quantitative pairwise progression	2.0	14.0	3.0	0.0
Symmetry*	5.0	0.0	0.0	0.0

0.94 for precision, 0.72 for recall and 0.82 for the F1 score. The lower number for recall indicates that we did not retrieve all rule tokens, however, as we show the agent problem-solving accuracy results below, it did not impact significantly the results generated during the solution prediction phase. We found that in most cases the rule tokens which were identified (94%) bore sufficient information for disambiguating the correct solution.

Figure 3.2 shows the detailed result of matching accuracy per Raven's set. We measured a ratio of accurately identified rules as *precision*, a ratio of all retrieved rules as *recall*, the geometric mean of the precision and recall as *f1 score*, and a ratio of overlapping rules between ground truth and induced rules as *overlap*. The algorithm achieved the overall precision of 94%, responding correctly on all problems in set C; and mismatching one problem from each, sets B, D and E. Using the Structural Affinity method, which reasons on the level of pixels and their interactions between images, the algorithm is able to infer the underlying logical relation in the Raven's problem with high precision. We will look at the rule response mismatch cases in the section that follows.

3.2.3 Error Analysis of the Rule Inference

In this section we examine cases where the logical relation is either not identified, or the choice is suboptimal. The accuracy results shown in Figure 3.2 are evaluated using *Carpenter's prior*, i.e. rules are not weighted identically or uniformly. The posterior probability of

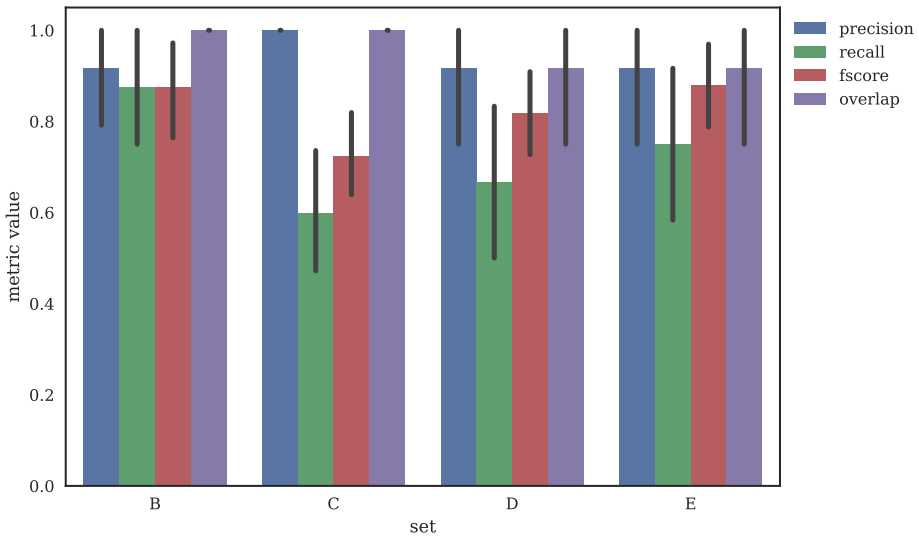


Figure 3.2: Evaluation of rule induction accuracy per Set for Raven's Standard Progressive Matrices Test - Set B, C, D and E

a rule is proportional to the frequency of occurrence with additional evidence provided by the model.

Figure 3.3 which implies a logical relation of subtraction, while not posing a notable difficulty for a human test taker, is challenging for the structural affinity algorithm. Fundamentally, the model is searching for the compatibility between image items in the Raven's problem. And, as such, a significant item misalignment in the corresponding images, leads to noise and subsequently incorrect response.

The logical relation of the Raven's problem shown in Figure 3.4 is *different completeness and different slope in each row and column* does not have a direct mapping to one of the rules in Carpenter's classification.

The third type of errors, which is only manifested with uniform prior rule probability, is caused by the ambiguity of the best fit.

Figure 3.3: An RPM image demonstrating vertical and horizontal subtraction with item misalignment

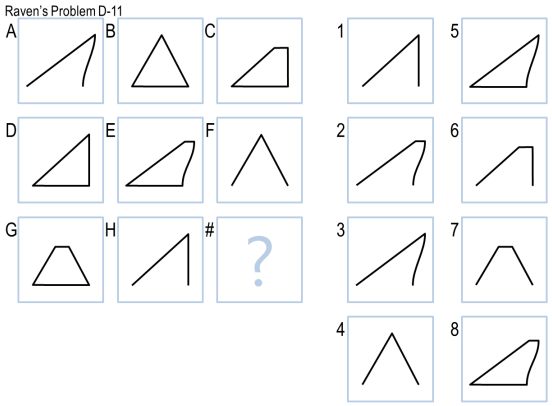


Figure 3.4: An RPM image demonstrating logical relation which does not fit into Carpenter's rule classification scheme

3.3 Rule-Aware Agent Results

3.3.1 Result Overview

This overview presents results achieved with a model that predicts a response given the most likely set of rules to produce the pattern in the given Raven's problem. We strategically focus on the subset of Standard Progressive Matrices, sets B through E, which represent a variety of 3x3 problems that allow extensive implementation of rule taxonomy described by P. Carpenter et. al., and a set of 2x2 problems for which we derive a similar

set of rules. [8] The proposed model with Structural Affinity algorithm predicts correct responses for 44 out of 48 targeted problems resulting in overall 91%. Table 3.2 and Figure 3.5 show the absolute count of the correct responses and its percentage per set.

The set C is the easiest of the three, and, not surprisingly, the agent can predict the correct responses given the previously inferred rule or a combination of rules. The prevailing rule for set C is the Quantitative pairwise progression (see Table 3.1) for which the model can accurately apply a set of corresponding heuristics to estimate a probability of the response.

The most common rule for set D is Distribution of three values, or the object permutation rule. Each of the three of the incorrectly predicted responses has a different reason, although, in principle, they all point to the limitation of the Structural Affinity method, and more specifically to the statistical and purely visual nature of the algorithm. The incorrect responses of those three problems can be mapped to two of four of the conceptual error types - Wrong Principle and Incomplete Correlate (the other two are Repetition and Difference) [21]. As Structural Affinity method is based on the image agreement, in a case of ambiguity it is biased towards selecting an answer which is a copy of the elements from the matrix space of the problem (Wrong Principle), or the answer is almost correct, i.e., second best choice (Incomplete Correlate).

The single incorrect response from the set E falls under the category of Repetition since the answer was chosen from an adjacent entry in the matrix space. Although if we consider the Repetition as ”perceptual matching between the matrix entries closest to the blank space and the available answers” interpretation, then the error from this set can be more likely attributed to the Incomplete Correlate since the model evaluates the influence from the entire row and/or column as opposed to adjacent cells only. The fact that choice fell on the answer which is also a copy is most likely due to chance.

Table 3.2: Response predictions for rule-aware agent per set for Raven's Standard Progressive Matrices Test - Set B, C, D and E

Set	**Number Correct**	**Percentage Correct**
Raven's Problem B	11	91.67
Raven's Problem C	12	100.0
Raven's Problem D	10	83.33
Raven's Problem E	11	91.67

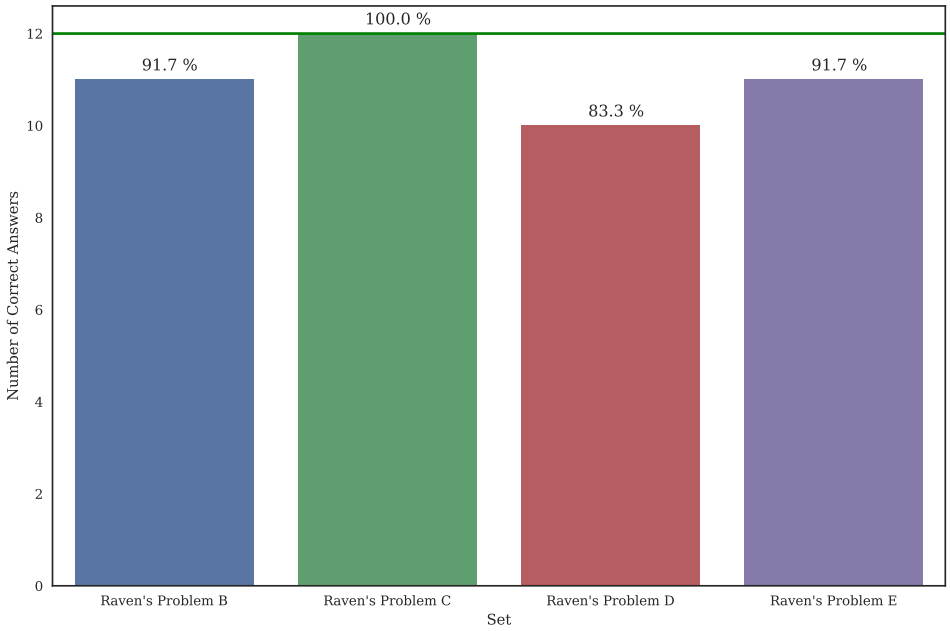


Figure 3.5: Evaluation of rule-aware agent's accuracy per set for Raven's Standard Progressive Matrices Test - Set B, C, D and E

3.3.2 Comparison to Other Computational Models

Figure 3.6 and extended results in Table 3.3 (with number of attempted problems and the resulting accuracy per set) show the comparison of our technique based on Structural Affinity method with two visual methods - Affine [22] and Fractal [10] and two propositional methods -Anthropomorphic [23] and CogSketch [9]. The Affine and the Fractal methods are using a visual approach and problem re-representation which relate to the Structural Affinity in Gestalt principle. The Anthropomorphic and the latest published CogSketch models differ from our method in the overall strategy, and as such also serve as good comparison models. The Structural Affinity method has a total score of 44 out of 48 targeted

Table 3.3: Comparison of the overall accuracy results for five computational models

Method	**Correct**	**Attempted**	**Correct(%)**
Affine	39	60	65.0
Antropomorphic	28	36	77.78
CogSketch	44	48	91.67
Fractal	42	60	70.0
Structural Affinity	44	48	91.67

problems in the SPM; the Affine and Fractal have total scores 39 and 42 respectively on the same problem set. The Anthropomorphic model solved 28 out of 36 targeted problems (C through E), and, finally, the CogSketch reported solving 44 out of 48 problems. The results presented in our Structural Affinity method compare very well in performance with CogSketch and Fractal methods and outperform Affine and Anthropomorphic accounts. Thus, we suggest that the method of first learning the pattern corresponding to Carpenter et al.'s rules as tokens, and then recognizing the answers that fit the pattern the best, plays an important role in constructing a model capable of solving intelligence tests in the form of geometric analogies.

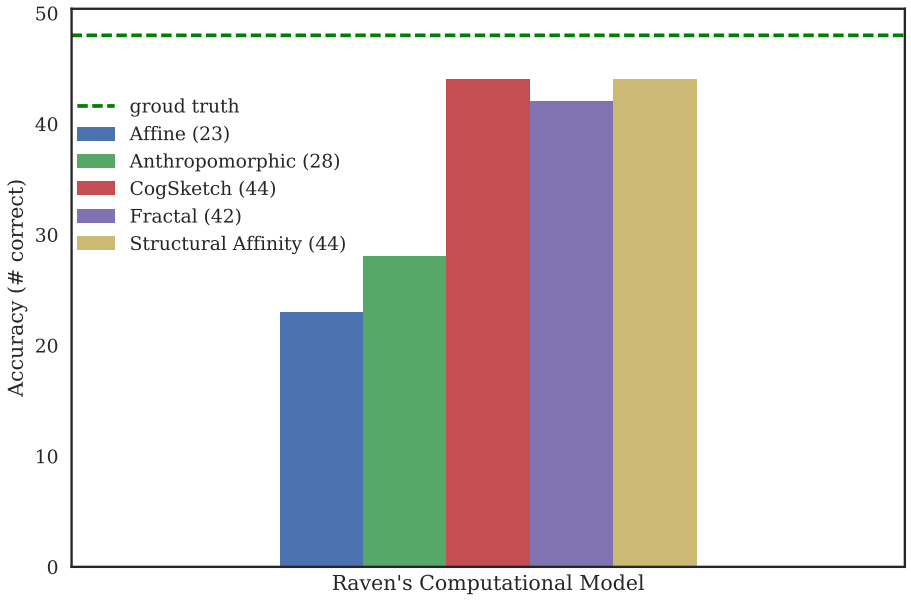


Figure 3.6: Comparison of the overall accuracy results for five computational models

CHAPTER 4
CONCLUSION

4.1 Discussion and Insights

We began by asserting the importance of learning and recognizing patterns in problem-solving by exploiting the interpretative nature of graphical models. We offered a representation - Markov Random Fields parameterized by affinity factors - that is capable of quantifying the level of interaction between constituents of the geometric problem on an example of Raven's intelligence test. The foundation of the discussed pattern learning algorithm is built on the premises of independence relations which determine the structure of the graph. Our key insight from the investigation of the inferred relationships between components of the images in Raven's problem is that a connection can be drawn between a set of common rules for solving Raven's intelligence test, such as those provided in Carpenter et. al's taxonomy, and the topologies of the graphical structures. We proposed a mapping between instances of learned structures, represented as a set of affinity factors, and a set of rule tokens that are regarded as strategies for correctly identifying the solution to the missing item in the problem. By encoding the interactions between images with affinity factors, we expressed the presence of a connection and its strength, both of which provided evidence to elicit a particular rule.

The simulation results presented in this work show that our Structural Affinity method built on the basis of Markov Networks allows inducing the set of rules that are most imaginable to express the logical sequence used for creating the problem. As our model generates input for structure finding algorithm directly from the images, it is not susceptible to the limitation of Carpenter et al. [8] and Little et al. [15] works where the model inputs are hand-coded.

The success of the rule inference portion of the algorithm (91% precision score) suggests that geometric problems on the Raven's test can be understood through statistical analyses that are also frequently leveraged in understanding other phenomena such as language and vision. The cases where the model misclassified the patterns are also difficult for humans as they require the analysis of slopes and completeness [19]. One of the problems, however, pointed to the limitation of our own approach where affinity factors could not capture the corresponding regions of the images. It may be possible to address this criticism by a more advanced factor-creation algorithm that can account for a more significant image misalignment.

Our second insight is a direct consequence of the proposed knowledge representation and the ability of the computational model to accurately predict the response by recognizing the learned pattern. The algorithm unfolds its reasoning by operating on a purely visual level through quantifying pixel state transitions and proceeds by creating higher levels of abstractions such as structures. This strongly supports the concept of a *visual thinking* thoroughly explored in the study of visual problem-solving in autism [22]. We assert that a task that involves utilizing logical deduction reasoning in geometric problems, does not require detecting objects in the presented problem. By achieving a high accuracy, our computational model demonstrates that it is not necessary to *know* what the objects are. Instead, our *visual* algorithm explores relationships between images and predicts responses which attempt to maintains the characteristic feature of the expected relationship.

An additional consequence of the second component of our algorithm - applying the induced rule tokens to Raven's problem to predict the solution - is a demonstration the heuristic nature of problem-solving through pattern recognition. Each inferred rule dictates the set of heuristics expressed through affinity factor states' relationships either validating or refusing a solution (we showed an example of an identity heuristic function). Our achieved accuracy of 91% is on par with the state of art computational accounts for solving SPM. The misclassified cases, most suitably attributed to a Wrong Principle or an Incom-

plete Correlate, suggest a potential improvement of the solution disambiguation by adding a more comprehensive set of heuristics. The generality of the model is supported by adhering to the standard set of rules as classified by Carpenter et al. [8] and applied to Bayesian models and evaluated to human performance data fitness [15].

Our method composed of three modules - representation building, pattern learning and pattern recognition - raises the question on the power of statistical reasoning for understanding problems that requiring logical deductions: what is the minimal level of abstraction that is sufficient for pattern extraction and accurate response prediction? The evidence shared with the results of our simulations suggests that high accuracy levels are indeed achievable with statistical reasoning over graphical models. Furthermore, as the underlying process of identifying structure is not specific to SPM, our method may be more general and applicable to other visual problems that require deducing logical sequences.

The compilation of results presented in this computational simulation provides evidence that Markov Network representation parameterized with *affinity* factor functions encode sufficient information for solving Raven's matrices of varying difficulty. By correctly answering 44 out of 48 problems, the methodology demonstrates an ability to match different levels of intelligence by strategizing over the Raven's matrix view. By formalizing the concept of compatibility between images, we believe it may be possible to generalize the approach to solving other geometrical problems that encompass mathematical symmetries and more advanced logical progressions.

4.2 Future Work

The most natural generalization of our method is expected for the Advanced Raven Progressive Matrices (APM) test due to their similar structure to SPM. The logical progressions are more complex so an addition of new objective functions may be required to capture the heuristics not observed on the simpler Raven's sets. An extension to Colored Raven Progressive Matrices (CPM) can be achieved by increasing the possible states of the affinity

factors from binary (black and white) to an arbitrary number of colors with the trade-off of computational speed.

Other geometrical problems, such as Odd One Out, have been addressed within the same family of visual computational models [7]. While further development is needed to validate the applicability of our approach here, we anticipate a minimal changes in the prediction algorithm (optimizing for lest compatible images), and a more substantial effort to infer a different set of rules to explain the patterns in the underlying problems.

An essential aspect of learning is being able to apply the past experiences to the new problems. The decisions of a human test-taker are typically more fluid in the presence of the instant feedback that changes the learning trajectory. The limitation of our model is that it assumes independence between problems, i.e., each problem is considered individually and no feedback is given to the agent. Given the probabilistic nature of our method (the responses are recorded by assigning a probability value to each candidate solution), it is feasible to estimate a *confidence* metric directly from the solution's score. The more immediate application of the confidence score is avoiding *guessing* the answer when reasoning was ambiguous or weak. Another use of the confidence computation is to serve as an instant feedback allowing the agent to auto-tune its algorithm if the confidence is insufficient [10]. To address the limitation of our model of bounding the space to a single problem, the confidence metric can influence the learning trajectory *across* problems. The scope of the Structural Affinity method is constrained to computing the compatibility between images *within* the problem. By extending the agent's hierarchy of estimating the similarity *between* problems, we can better approximate the human cognition of solving a problem by analogy. This approximation becomes attainable as we *learn* to understand images and uncover their intended meaning by analyzing discoverable structures that provide interpretation of the main features of the given context.

Appendices

APPENDIX A

DATA PROCESSING

An individual problem in the Raven's test is represented with two sets of images: matrix space $\{S_{3x3} : I_A..I_G\}$ and a solution space $\{S_{3x3} : I_1..I_8\}$, or for a 2x2 case $\{S_{2x2} : I_A..I_C\}$ and $\{S_{2x2} : I_1..I_6\}$. Each image is processed with Python open-source library *sklearn* which supports a variety of algorithms for image handling and transformations [24]. In this work, we use a basic loading and transformation procedure to represent an image with dimensions $N \times M$ as a binary array of size $N \times M$. To accomplish this, the original image is converted first to a gray-scale, where each pixel carries only intensity information. This reduces the image from three-dimensional representation (with the third being the color), to a two-dimensional matrix. In addition to gray scaling, we further simplify the image representation by *binarizing* it, i.e. each pixel can take one of the two values - 0 (white) or 1 (pure black). This makes the affinity factor generation very simple as we need to create only 2^D configuration combinations, where D is the scope size of the affinity factor. We perform binarization by choosing a threshold and setting each pixel above it to 1, otherwise leaving it as 0. The overall image transformation can be expressed via chain A.1:

$$f : I \rightarrow Gray(I) \rightarrow \mathbf{1}(x) = \begin{cases} 1, & \text{if x} \geq t \\ 0, & \text{if x ¡ t} \end{cases} \tag{A.1}$$

where threshold t is selected as a mean value of the pixel intensities:

$$t = \frac{\sum_{n=1}^{N \times M} Intensity}{N \times M} \tag{A.2}$$

And, finally, we flatten the binary matrix $N \times M$ into a vector of length $N \times M$ to compute pairwise image compatibility.

APPENDIX B

HEURISTIC FUNCTIONS FOR RULE DISCOVERY

The response prediction phase received the problem encoded with affinity factor representation, and a set of token rule induced in during the rule learning step. The Table B.1 details the map between the rule and its objective function that is either maximized or minimized.

Table B.1: Table to map rules to corresponding objective functions used during the response prediction phase

Rule	Objective Function	Description
Constant	$max \quad F = \phi_{0,0}(I_1, I_2)$	Maximize the potential of the black pixel configuration
	$min \quad F = \phi_{0,1}(I_1, I_2)$	Minimize the potential of transition from black to white pixel
	$min \quad F = \phi_{1,0}(I_1, I_2)$	Minimize the potential of transition from white to black pixel
Quantitative	Increasing: $max \; F = \phi_{0,0,0}(I_1, I_2, I_3)$ & $min \; F = \phi_{0,1}(I_1, I_2) - \phi_{1,0}(I_2, I_3)$	Maximize the configuration where both the number of black pixels is highest and Minimize the configuration where number of black pixels in the third image is decreasing

Pairwise Progression	Decreasing: $max \quad F = \phi_{0,1}(I_1, I_2) - \phi_{1,0}(I_2, I_3)$	Same as above except flip the second objective function to maximization
Figure Addition	$max \quad F = \phi_{0,0}(I_1, I_2)$	Maximize the potential of the black pixel configuration
	$min \quad F = \phi_{1,0}(I_{11}, I_{12}) - \phi_{1,0}(I_{21}, I_{22})$	Minimize the difference between expected transition of white to black ratio and the observed ratio
Figure Subtraction	$min \quad F = \phi_{1,0}(I_1, I_2)$	Minimize the potential of the white to black pixel configuration
Subtraction/Boolean AND	$max \quad F = \phi_{0,0,0}(I_1, I_2, I_3)$	Maximize the configuration of the black pixel potential
	$min \quad F = \phi_{0,1,0}(I_1, I_2, I_3)$	Minimize a configuration which represents invalid conjunction result
Addition/Boolean OR	$max \quad F = \phi_{1,0,0}(I_1, I_2, I_3)$	Maximize a configuration which reflect the summation operator
	$min \quad F = \phi_{1,1,0}(I_1, I_2, I_3)$	Minimize a configuration which represents invalid summation result

Distribution of two	$max \quad F = \phi_{0,0,1}(I_1, I_2, I_3)$	Maximize a configuration where the third image disagrees with first two by flipping to the white state
	$max \quad F = \phi_{1,1,0}(I_1, I_2, I_3)$	Maximize a configuration where the third image disagrees with first two by flipping to the black state

REFERENCES

[1] G. Polya, *How to solve it: A new aspect of mathematical model*, 1945.

[2] A. Newell, "The heuristic of george polya and its relation to artificial intelligence," *Methods of heuristics*, pp. 195–243, 1983.

[3] S. Bringsjord, "Psychometric artificial intelligence," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 23, no. 3, pp. 271–277, 2011.

[4] J. Hernández-Orallo, F. Martínez-Plumed, U. Schmid, M. Siebers, and D. L. Dowe, "Computer models solving intelligence test problems: Progress and implications," *Artificial Intelligence*, vol. 230, pp. 74–107, 2016.

[5] T. G. Evans, "A heuristic program to solve geometric-analogy problems," in *Proceedings of the April 21-23, 1964, spring joint computer conference*, ACM, 1964, pp. 327–338.

[6] A. Lovett, K. Lockwood, and K. Forbus, "A computational model of the visual oddity task," in *The Proceedings of the 30th Annual Conference of the Cognitive Science Society. Washington, DC*, vol. 25, 2008, p. 29.

[7] K. McGreggor and A. Goel, "Finding the odd one out: A fractal analogical approach," in *Proceedings of the 8th ACM conference on Creativity and cognition*, ACM, 2011, pp. 289–298.

[8] P. A. Carpenter, M. A. Just, and P. Shell, "What one intelligence test measures: A theoretical account of the processing in the raven progressive matrices test.," *Psychological review*, vol. 97, no. 3, p. 404, 1990.

[9] A. Lovett, K. Forbus, and J. Usher, "A structure-mapping model of raven's progressive matrices," in *Proceedings of the Cognitive Science Society*, vol. 32, 2010.

[10] K. McGreggor and A. K. Goel, "Confident reasoning on raven's progressive matrices tests.," in *AAAI*, 2014, pp. 380–386.

[11] C. Strannegård, S. Cirillo, and V. Ström, "An anthropomorphic method for progressive matrix problems," *Cognitive Systems Research*, vol. 22, pp. 35–46, 2013.

[12] E. Hunt, "Quote the raven? nevermore.," 1974.

[13] K. McGreggor, M. Kunda, and A. K. Goel, "A fractal analogy approach to the raven's test of intelligence.," in *Visual Representations and Reasoning*, 2010.

[14] M. Ragni and S. Neubert, "Analyzing ravens intelligence test: Cognitive model, demand, and complexity," in *Computational Approaches to Analogical Reasoning: Current Trends*, Springer, 2014, pp. 351–370.

[15] D. R. Little, S. Lewandowsky, and T Griffiths, "A bayesian model of rule induction in ravens progressive matrices," in *Proceedings of the 34th annual conference of the cognitive science society*, 2012, pp. 1918–1923.

[16] D. Koller and N. Friedman, *Probabilistic graphical models: Principles and techniques*. MIT press, 2009.

[17] J. C. Raven and J. H. Court, *Ravens progressive matrices and vocabulary scales*. Oxford Psychologists Press Oxford, UK, 1998.

[18] L. B. V. Matzen, M. W. Van der Molen, and A. C. Dudink, "Error analysis of raven test performance," *Personality and Individual Differences*, vol. 16, no. 3, pp. 433–445, 1994.

[19] N. Georgiev, "Item analysis of c, d and e series from raven's standard progressive matrices with item response theory two-parameter logistic model," *Europe's Journal of Psychology*, vol. 4, no. 3, 2008.

[20] M. Kunda, I. Soulières, A. Rozga, and A. K. Goel, "Error patterns on the raven's standard progressive matrices test," *Intelligence*, vol. 59, pp. 181–198, 2016.

[21] J Raven, "Court jh. 2003. manual for raven's progressive matrices and vocabulary scales," *Section1: General Overview. San Antonia: Harcourt Assessment*,

[22] M. Kunda, "Visual problem solving in autism, psychometrics, and ai: The case of the raven's progressive matrices intelligence test," 2013.

[23] S Cirillo and V Ström, "An anthropomorphic solver for raven's progressive matrices (no. 2010: 096)," *Goteborg, Sweden: Chalmers University of Technology*, 2010.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.